

Soft Body Dynamics

Jungdam Won

Computer Science & Engineering
Seoul National Univ.

This material was created based on the slides and lecture notes of *Physically Based Modeling* (SIGGRAPH 2001 course) by Andrew Witkin and *Dynamic Deformables: Implementation and Production Practicalities (Now With Code!)* by Ted Kim

Where are we?

- We have learned about so far
 - How to simulate a single particle
 - How to simulate multiple particles
 - How to simulate multiple particles with soft constraints (e.g. spring)
 - How to simulate multiple particles with hard constraints (e.g. a bead on wire)
 - How to integrate mass-spring systems in a more stable manner (i.e. implicit integration)

Soft Body Objects

- An object of which shape can be *deformed* when external forces are applied
 - Softness is a measure of the degree of deformation, the softer the material is, the more it deforms
- When external force is removed, the shape of object will be *recovered* (to some extent)

Hyperelastic Deformation

- Objects in reality will be torn out when too large external forces are applied, or they could be deformed permanently due to the irreversible change of internal structure, which is called ***plastic deformation***
- In this lecture, we will only consider ***hyperelastic deformation***, where the objects will be perfectly recovered to their original shapes when all external forces are removed

Examples of Soft Body Objects



<https://www.myrecipes.com/dessert-recipes/how-to-update-molded-gelatin-jello-desserts>

Best Hair Extensions For Fine Hair | Sitting Pretty (sittingprettyhalohair.com)

Scotch-Brite® Dusting Cloth | 3M United States

The Secret Numbers On Your Tyres And What They Mean – Carpockets

Dynamic Deformables:Implementation and Production Practicalities (SIGGRAPH 2022 Course)

Essential of Soft Body Dynamics

- Simulating soft body objects are all about measuring and recovering ***deformation***, where we should answer two questions:

- 1. How far am I from my original shape?***

More concretely, how can I compute a quantitative deformation score that tells me exactly how stretched or squashed I am, relative to my original shape?

- 2. How much should I push back?***

Once I know how extactly how deformed I am, what should I do with this knowledge?

Essential of Soft Body Dynamics

- Simulating soft body objects are all about measuring and recovering ***deformation***, where we should answer two questions:

1. ***How far am I from my original shape?***

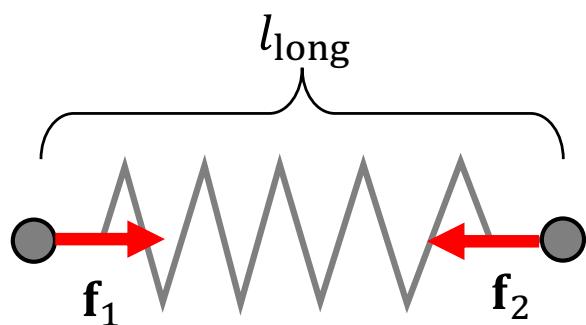
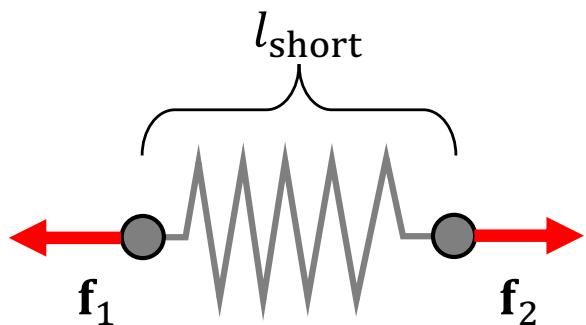
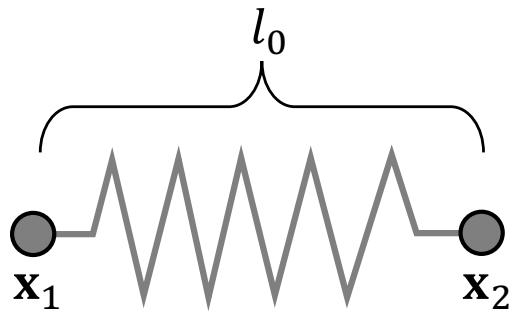
$$\Psi(\mathbf{X})$$

where \mathbf{X} is the state vector and $\Psi > 0$

2. ***How much should I push back?***

$$\mathbf{f} \propto -\frac{\partial \Psi(\mathbf{X})}{\partial \mathbf{X}}$$

An Example (Linear Spring)



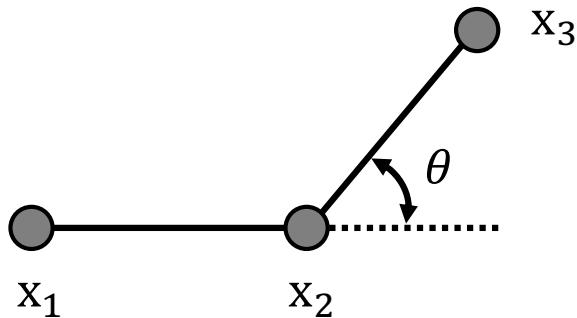
$$\Psi(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{2} k(|\mathbf{l}| - l_0)^2$$

where $\mathbf{l} = \mathbf{x}_1 - \mathbf{x}_2$

$$\mathbf{f}_1 = -\frac{\partial \Psi}{\partial \mathbf{x}_1} = -\frac{\partial \Psi}{\partial \mathbf{l}} \frac{\partial \mathbf{l}}{\partial \mathbf{x}_1} = -k(|\mathbf{l}| - l_0) \frac{\mathbf{l}}{|\mathbf{l}|}$$

$$\mathbf{f}_2 = -\mathbf{f}_1$$

An Example (Angular Spring)



$$\Psi(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \frac{1}{2} k \theta^2$$

where $\theta = \cos^{-1}(\mathbf{l}_1 \cdot \mathbf{l}_2)$

$$\mathbf{l}_1 = \frac{\mathbf{x}_2 - \mathbf{x}_1}{|\mathbf{x}_2 - \mathbf{x}_1|}, \mathbf{l}_2 = \frac{\mathbf{x}_3 - \mathbf{x}_2}{|\mathbf{x}_3 - \mathbf{x}_2|}$$

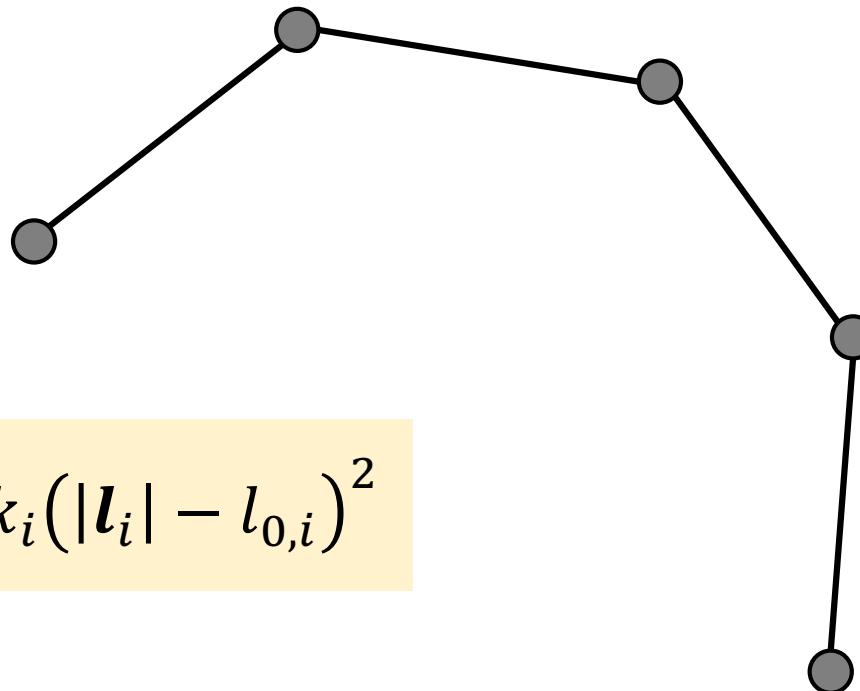
$$\mathbf{f}_1 = -\frac{\partial \Psi}{\partial \mathbf{x}_1} = -\frac{\partial \Psi}{\partial \theta} \frac{\partial \theta}{\partial \mathbf{l}_1} \frac{\partial \mathbf{l}_1}{\partial \mathbf{x}_1} = -k\theta \frac{\partial \theta}{\partial \mathbf{l}_1} \frac{\partial \mathbf{l}_1}{\partial \mathbf{x}_1}$$

$$\mathbf{f}_2 = -k\theta \frac{\partial \theta}{\partial \mathbf{x}_2}$$

$$\mathbf{f}_3 = -k\theta \frac{\partial \theta}{\partial \mathbf{x}_3}$$

Hair Model

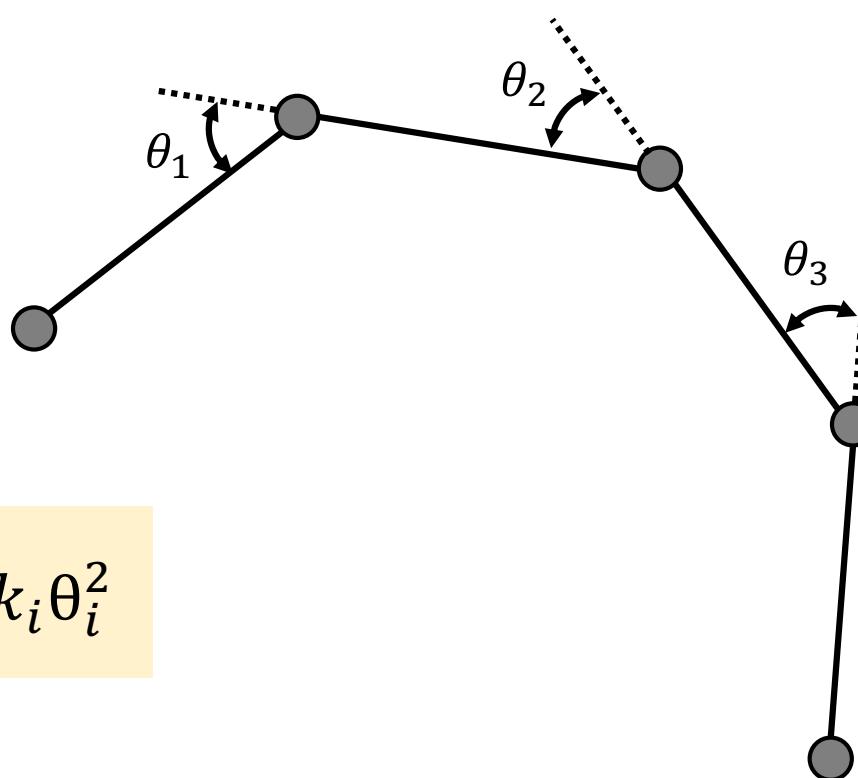
- Limp hair: Just a set of springs



$$\Psi = \frac{1}{2} \sum k_i (|l_i| - l_{0,i})^2$$

Hair Model

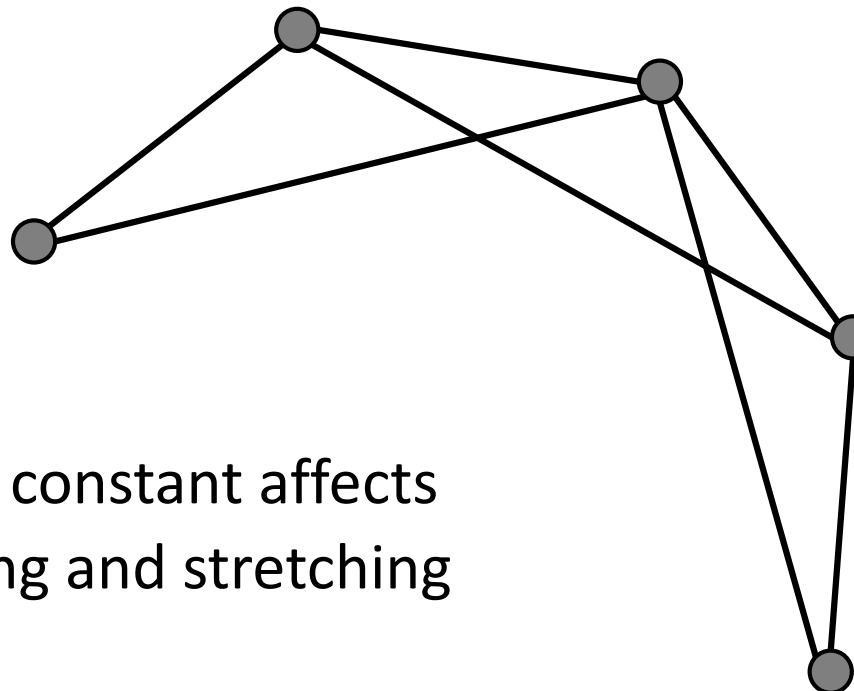
- Stiff hair: Add angular springs



$$\Psi = \frac{1}{2} \sum k_i \theta_i^2$$

Hair Model

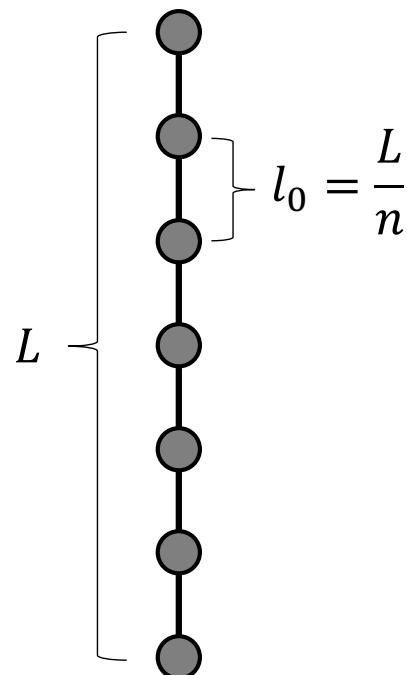
- Alternative: More linear springs



- Difficulty:
Each spring constant affects
both bending and stretching

Discretization

- Make sure energy independent of sampling



Total energy:

$$\Psi = \frac{1}{2} k \sum (l - l_0)^2$$

Stretch 100%:

$$\Psi = \frac{1}{2} k \sum \left(\frac{L}{n}\right)^2 = \frac{1}{2} \frac{k}{n} L^2$$

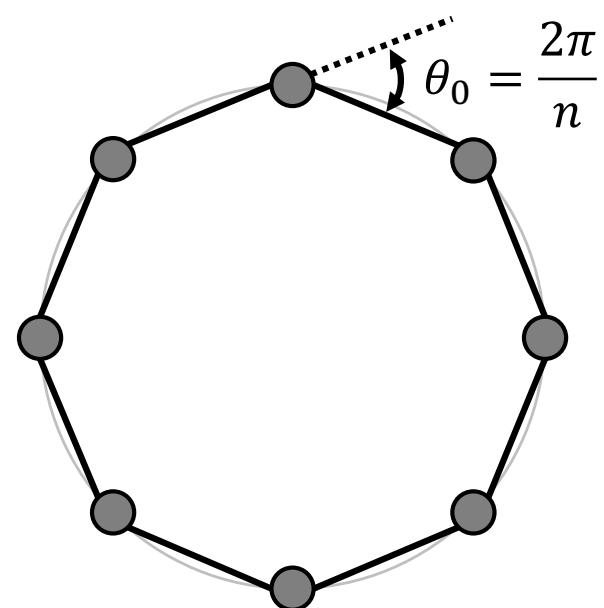
Constant energy implies:

$$k \propto n$$

Note: Higher sampling (discretization) \rightarrow more stiffness

Discretization

- Make sure energy independent of sampling



Total energy:

$$\Psi = \frac{1}{2} k \sum (\theta - \theta_0)^2$$

Straighten out 100%: $\Psi = \frac{1}{2} k \sum \left(\frac{2\pi}{n}\right)^2 = \frac{1}{2} \frac{k}{n} (2\pi)^2$

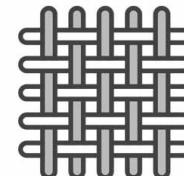
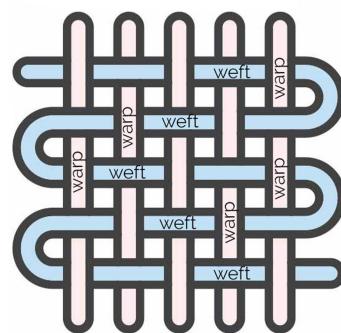
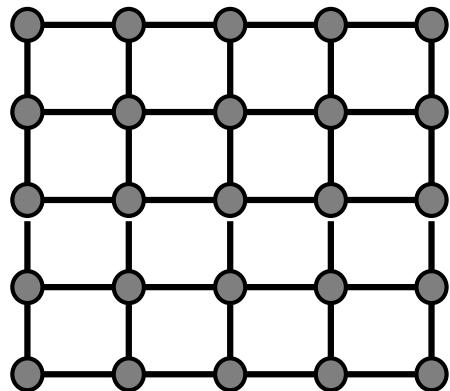
Constant energy implies:

$$k \propto n$$

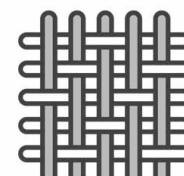
Note: Higher sampling (discretization) \rightarrow more stiffness

Clothing

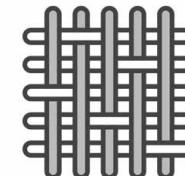
- Start with warp and weft threads
- Weave them together
- Add angular springs so that threads stay perpendicular each other



PLAIN WEAVE



TWILL WEAVE



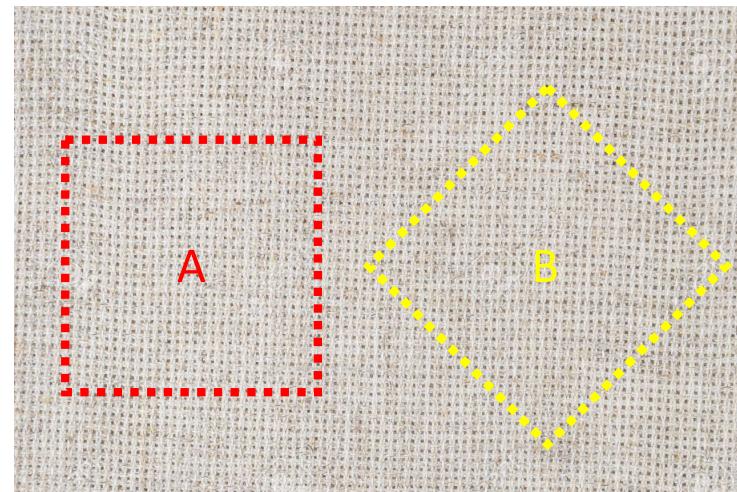
SATIN WEAVE



HERRINGBONE WEAVE

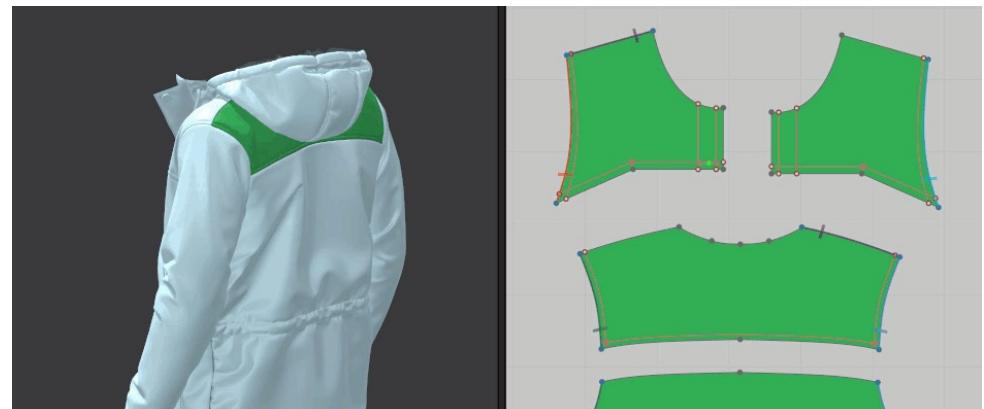
Things to Consider in Modeling Clothes

- Cloth resists
 - Stretching
 - Shearing
 - Bending
- Ward and Weft directions are special, A and B will move differently



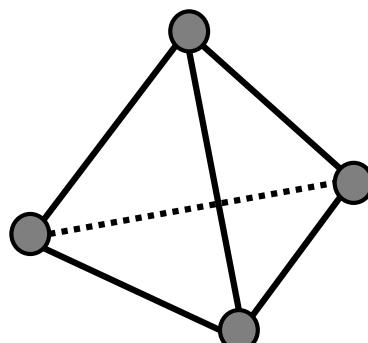
Rest Mesh Options

- Modeling in 3D
 - Clothing already on characters
 - Can directly craft desired 3D shape
 - Annotate warp/weft directions
 - Clothing probably will not locally flatten
- Modeling in 2D
 - Must put clothing on characters
 - Might need to hire a tailor to get the pattern right
 - Guaranteed to flatten locally
 - Sew parts together
 - Greater realism



Beyond Mass Spring System

- Pros
 - Easy to understand and implement
- Cons
 - Given 3D meshes, it is hard to setup springs that generate desired behaviors
 - To achieve desired behaviors, it typically requires too many constraints (i.e. over-constrained)



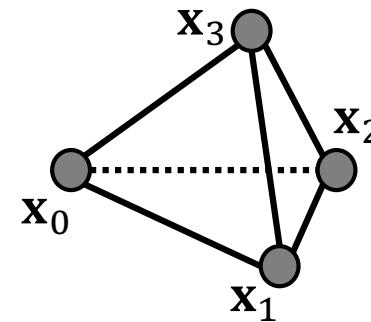
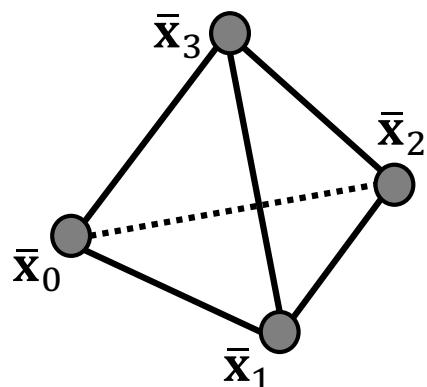
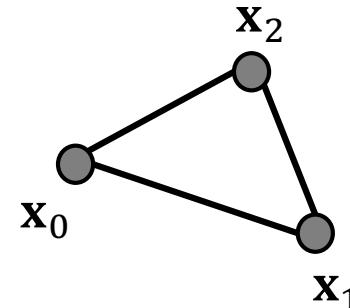
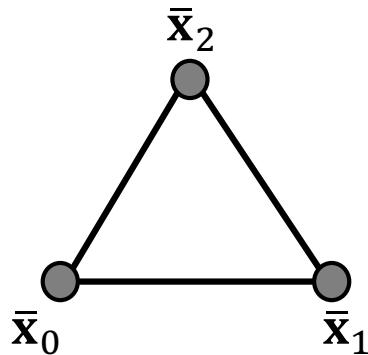
A tetrahedron in 3D

Degree-of-freedom: 12

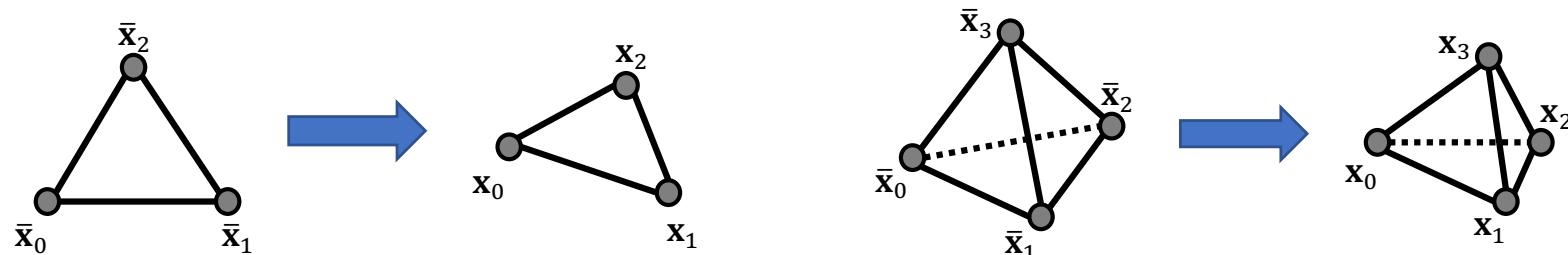
Constraints: 18 (linear/angular springs)

Beyond Mass Spring System

- **Idea:** Measure deformation in the level of basic elements (e.g., triangle, tethrahedron)



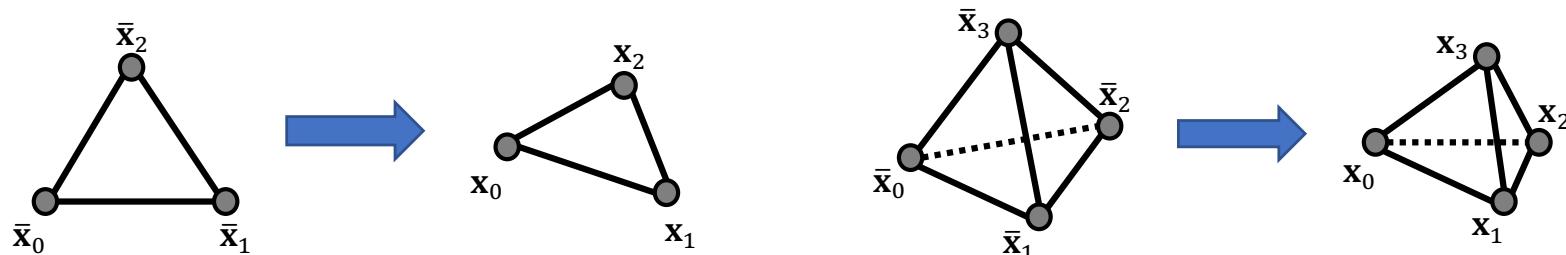
Measuring Deformation



$$\Psi_{\text{wrong}} = \sum \|\bar{\mathbf{x}}_i - \mathbf{x}_i\|_2$$

- It is sensitive to rigid transformation
- An ideal deformation score should be invariant to translation and rotation

Removing Translation

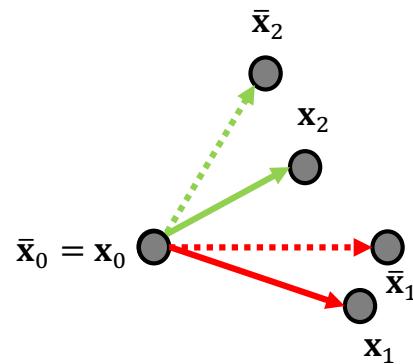


$$\phi(\bar{x}_i) = F\bar{x}_i + t = x_i \quad \text{Compute affine map}$$

$$\frac{\partial \phi(\bar{x}_i)}{\partial \bar{x}_i} = F$$

The matrix F is called the ***deformation gradient***, which will be the base to define following deformation score (Any deformation score depending only F will be translation-invariant)

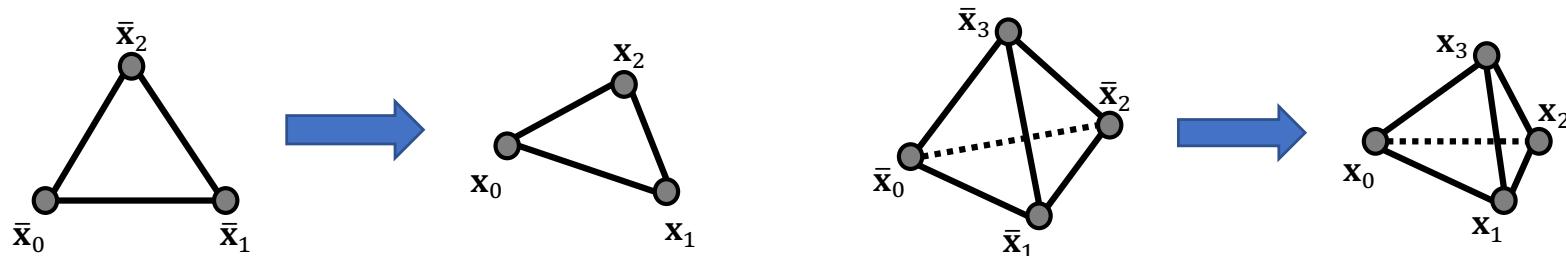
Computing F



$$\frac{\left(\mathbf{x}_1 - \mathbf{x}_0 \mid \mathbf{x}_2 - \mathbf{x}_0 \right)}{D_s} = F \frac{\left(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_0 \mid \bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_0 \right)}{D_m}$$

$$F = D_s D_m^{-1}$$

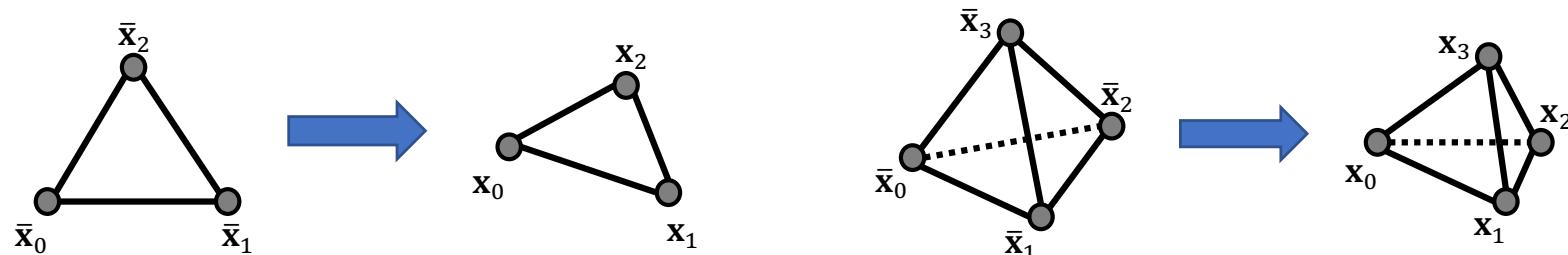
Measuring Deformation



$$\Psi_{\text{Dirichlet}} = \|F\|_F^2 = \sum \sum f_{ij}^2$$

- It is invariant to translation, however, it is still not invariant to rotation
- When F is the identity matrix, it returns non-zero
- When F is a zero matrix, it returns zero

Measuring Deformation

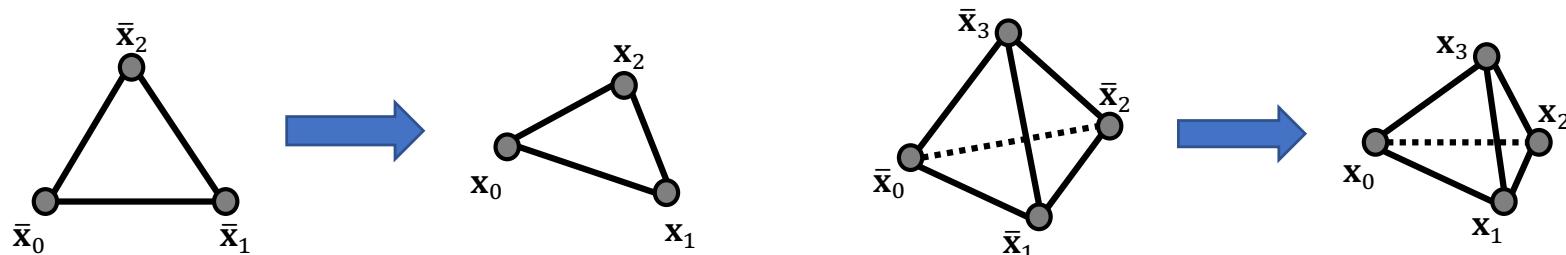


$$\Psi_{\text{StVK}} = \frac{1}{2} \| \mathbf{F}^T \mathbf{F} - \mathbf{I} \|_F^2$$

StVK: St. Venant-Kirchhoff

- When \mathbf{F} is a pure rotation, it gives us zero
- It is 4th-order in \mathbf{F} , which adds additional computational complexity

Measuring Deformation



$$\Psi_{\text{ARAP}} = \frac{1}{2} \|F - R\|_F^2$$

ARAP : as rigid as possible

where $F = RS$ by polar decomposition

- When F is a pure rotation, it gives us zero
- It is at most quadratic
- R introduces additional difficulty in deriving $\frac{\partial \Psi_{\text{ARAP}}}{\partial x}$

Summary of Deformation Energy

Energy	Pros	Cons
Wrong	Not exist	Everything
Dirichlet	Not many	Doesn't measure deformation effectively, only invariant to translation
StVK	Returns zero when F is a pure rotation, and reasonable scores otherwise	Overly non-linear (quartic)
ARAP	Returns zero when F is a pure rotation, and reasonable scores otherwise, and is quadratic	That R term is going to cause trouble

Computing Forces

- To get the forces \mathbf{f} on the nodes of a triangle in 2D, we stack the vertices of our triangle $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$ into a big vector, and then take the gradient

$$\mathbf{f} = -a \frac{\partial \Psi}{\partial \mathbf{x}} \quad \mathbf{x}_1 = \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix} \in \mathcal{R}^6$$

- The same strategy also applies to other basic elements such as tetrahedrons

Computing Forces

$$\Psi_{\text{Dirichlet}} = \|F\|_F^2 = \sum \sum f_{ij}^2$$

$$\|F\|_F^2 = \text{tr}(F^T F)$$

$$f = -a \frac{\partial \Psi}{\partial x} = -a \frac{\partial \Psi_{\text{Dirichlet}}}{\partial x}$$

- Plug F in to $\Psi_{\text{Dirichlet}}$
- Multiply everything through to get a massive one-line equation for $\|F\|_F^2$
- Take the derivative of this massive equation six times, once for each entry in x , and stack the results into a force vector (while hoping you didn't make a mistake in your derivation somewhere)

Computing Forces

$$\Psi_{\text{Dirichlet}} = \|\mathbf{F}\|_F^2 = \sum \sum f_{ij}^2$$

$$\|\mathbf{F}\|_F^2 = \text{tr}(\mathbf{F}^T \mathbf{F})$$

$$\mathbf{f} = -a \frac{\partial \Psi}{\partial \mathbf{x}} = -a \frac{\partial \Psi_{\text{Dirichlet}}}{\partial \mathbf{x}}$$

$$\mathbf{F} = \mathbf{D}_s \mathbf{D}_m^{-1} = \left(\mathbf{x}_1 - \mathbf{x}_0 \middle| \mathbf{x}_2 - \mathbf{x}_0 \right) \left(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_0 \middle| \bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_0 \right)^{-1} = \left(\mathbf{x}_1 - \mathbf{x}_0 \middle| \mathbf{x}_2 - \mathbf{x}_0 \right) \begin{pmatrix} m_0 & m_2 \\ m_1 & m_3 \end{pmatrix}$$

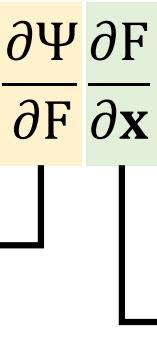
$$\mathbf{F}^T \mathbf{F} = \begin{pmatrix} m_0 & m_1 \\ m_2 & m_3 \end{pmatrix} \begin{pmatrix} (\mathbf{x}_1 - \mathbf{x}_0)^T (\mathbf{x}_1 - \mathbf{x}_0) & (\mathbf{x}_2 - \mathbf{x}_0)^T (\mathbf{x}_1 - \mathbf{x}_0) \\ (\mathbf{x}_2 - \mathbf{x}_0)^T (\mathbf{x}_1 - \mathbf{x}_0) & (\mathbf{x}_2 - \mathbf{x}_0)^T (\mathbf{x}_2 - \mathbf{x}_0) \end{pmatrix} \begin{pmatrix} m_0 & m_2 \\ m_1 & m_3 \end{pmatrix}$$

$$\begin{aligned} \|\mathbf{F}\|_F^2 &= m_0^2 (\mathbf{x}_1 - \mathbf{x}_0)^T (\mathbf{x}_1 - \mathbf{x}_0) + 2m_0 m_1 (\mathbf{x}_2 - \mathbf{x}_0)^T (\mathbf{x}_1 - \mathbf{x}_0) + m_1^2 (\mathbf{x}_2 - \mathbf{x}_0)^T (\mathbf{x}_2 - \mathbf{x}_0) \\ &\quad + m_2^2 (\mathbf{x}_1 - \mathbf{x}_0)^T (\mathbf{x}_1 - \mathbf{x}_0) + 2m_2 m_3 (\mathbf{x}_2 - \mathbf{x}_0)^T (\mathbf{x}_1 - \mathbf{x}_0) + m_3^2 (\mathbf{x}_2 - \mathbf{x}_0)^T (\mathbf{x}_2 - \mathbf{x}_0) \end{aligned}$$

$$\frac{\partial \|\mathbf{F}\|_F^2}{\partial \mathbf{x}} = \dots \quad \text{Computing gradients seem messy even for this simplest energy, and things will be getting worse as we use more complex energies}$$

Computing Forces (Better Way)

$$\mathbf{f} = -a \frac{\partial \Psi}{\partial \mathbf{x}} = -a \frac{\partial \Psi}{\partial \mathbf{F}} \frac{\partial \mathbf{F}}{\partial \mathbf{x}} = -a \frac{\partial \mathbf{F}}{\partial \mathbf{x}} : \frac{\partial \Psi}{\partial \mathbf{F}}$$

Matrix  Double-contraction

3rd-order tensor

$$\mathbf{A} : \mathbf{B} = \begin{bmatrix} a_0 & a_2 \\ a_1 & a_3 \end{bmatrix} \begin{bmatrix} b_0 & b_2 \\ b_1 & b_3 \end{bmatrix} = a_0 b_0 + a_1 b_1 + a_2 b_2 + a_3 b_3$$

$$\mathbf{A} : \mathbf{B} = \begin{bmatrix} \begin{bmatrix} a_0 & a_2 \\ a_1 & a_3 \end{bmatrix} \\ \begin{bmatrix} a_4 & a_6 \\ a_5 & a_7 \end{bmatrix} \\ \begin{bmatrix} a_8 & a_{10} \\ a_9 & a_{11} \end{bmatrix} \end{bmatrix} \begin{bmatrix} b_0 & b_2 \\ b_1 & b_3 \end{bmatrix} = \begin{bmatrix} a_0 b_0 + a_1 b_1 + a_2 b_2 + a_3 b_3 \\ a_4 b_0 + a_5 b_1 + a_6 b_2 + a_7 b_3 \\ a_8 b_0 + a_9 b_1 + a_{10} b_2 + a_{11} b_3 \end{bmatrix}$$

Computing Forces (Better Way)

$$\frac{\partial \mathbf{F}}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}} \mathbf{D}_s \mathbf{D}_m^{-1} = \left(\frac{\partial}{\partial \mathbf{x}} \mathbf{D}_s \right) \mathbf{D}_m^{-1}$$

$$\frac{\partial \mathbf{D}_s}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{D}_s}{\partial x_0} \\ \frac{\partial \mathbf{D}_s}{\partial x_1} \\ \vdots \\ \frac{\partial \mathbf{D}_s}{\partial x_5} \end{bmatrix} \quad \begin{aligned} \frac{\partial \mathbf{D}_s}{\partial x_0} &= \begin{bmatrix} -1 & -1 \\ 0 & 0 \end{bmatrix} & \frac{\partial \mathbf{D}_s}{\partial x_1} &= \begin{bmatrix} 0 & 0 \\ -1 & -1 \end{bmatrix} & \frac{\partial \mathbf{D}_s}{\partial x_2} &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \\ \frac{\partial \mathbf{D}_s}{\partial x_3} &= \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} & \frac{\partial \mathbf{D}_s}{\partial x_4} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} & \frac{\partial \mathbf{D}_s}{\partial x_5} &= \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned}$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_5 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_5 \end{bmatrix} \quad \mathbf{D}_s = \left(\mathbf{x}_1 - \mathbf{x}_0 \mid \mathbf{x}_2 - \mathbf{x}_0 \right) = \begin{pmatrix} x_2 - x_0 & x_4 - x_0 \\ x_3 - x_1 & x_5 - x_1 \end{pmatrix}$$

The derivative of the deformation gradient $\frac{\partial \mathbf{F}}{\partial \mathbf{x}}$ is **constant**
(it only depends on the type of base element)

Computing Forces (Better Way)

$$\Psi_{\text{Dirichlet}} = \|F\|_F^2 = \sum \sum f_{ij}^2$$

$$\|F\|_F^2 = \text{tr}(F^T F)$$

$$f = -a \frac{\partial \Psi_{\text{Dirichlet}}}{\partial x} = -a \frac{\partial F}{\partial x} \cdot \frac{\partial \Psi_{\text{Dirichlet}}}{\partial F}$$

$$\frac{\partial F}{\partial x} = \begin{bmatrix} [-1 & -1] \\ [0 & 0] \\ [0 & 0] \\ [-1 & -1] \\ [1 & 0] \\ [0 & 0] \\ [0 & 0] \\ [1 & 0] \\ [0 & 1] \\ [0 & 0] \\ [0 & 0] \\ [0 & 1] \end{bmatrix}$$

$$\frac{\partial \Psi_{\text{Dirichlet}}}{\partial F} = \frac{\partial \|F\|_F^2}{\partial F} = 2F$$

Other Energies

$$\Psi_{\text{StVK}} = \frac{1}{2} \|F^T F - I\|_F^2$$

$$\Psi_{\text{ARAP}} = \frac{1}{2} \|F - R\|_F^2$$

$$\frac{\partial \Psi_{\text{StVK}}}{\partial x} = F(F^T F - I)$$

$$\frac{\partial \Psi_{\text{ARAP}}}{\partial x} = F - R$$

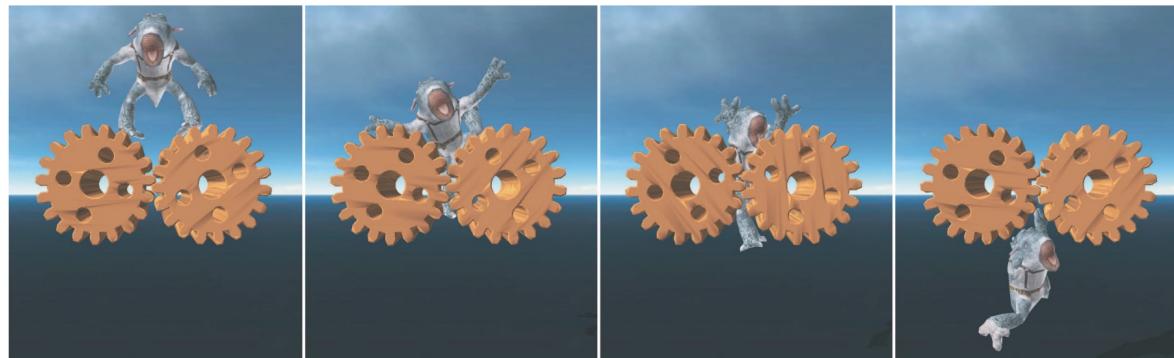
- You can verify this at home, or refer to the details in the lecture note below

Other Formulations for Simulating Soft Bodies

- **Position-based Dynamics**
- Projective Dynamics
- Material Point Method

Position-based Dynamics

- It was first introduced by Matthias Müller in 2006
- As the name implies, the basic idea is to formulate dynamics in the level of position (+ velocity) only without forces
- It focuses on fast computation and stability



PBD Algorithm

while simulating

for all particles i

$$\mathbf{v}_i \leftarrow \mathbf{v}_i + h\mathbf{f}_{\text{ext}}(\mathbf{x}_i)$$

$$\mathbf{p}_i \leftarrow \mathbf{x}_i$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + h\mathbf{v}_i$$

for all constraints c

 solve(c, h)

for all particles i

$$\mathbf{v}_i \leftarrow (\mathbf{x}_i - \mathbf{p}_i)/h$$

solve(c, h):

for all particles i of c

 compute $\Delta\mathbf{x}_i$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta\mathbf{x}_i$$

Iteration vs. Sub-steps

while simulating

for all particles i

$$\mathbf{v}_i \leftarrow \mathbf{v}_i + h\mathbf{f}_{\text{ext}}(\mathbf{x}_i)$$

$$\mathbf{p}_i \leftarrow \mathbf{x}_i$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + h\mathbf{v}_i$$

for n iterations

for all constraints c

 solve(c , h)

for all particles i

$$\mathbf{v}_i \leftarrow (\mathbf{x}_i - \mathbf{p}_i)/h$$

while simulating

for n iterations

for all particles i

$$\mathbf{v}_i \leftarrow \mathbf{v}_i + h\mathbf{g}$$

$$\mathbf{p}_i \leftarrow \mathbf{x}_i$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + h\mathbf{v}_i$$

for all constraints c

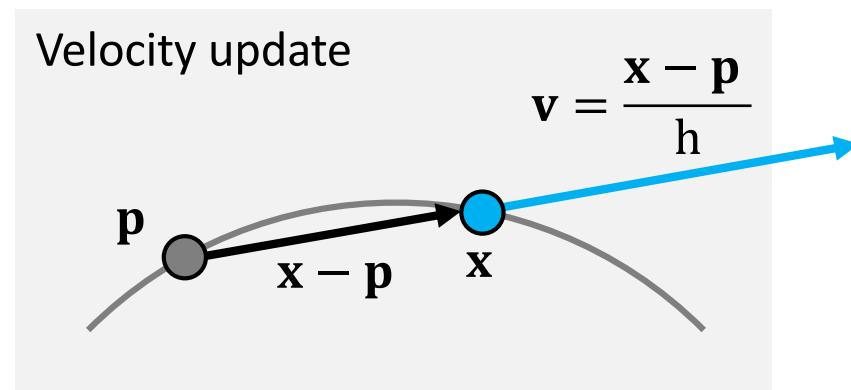
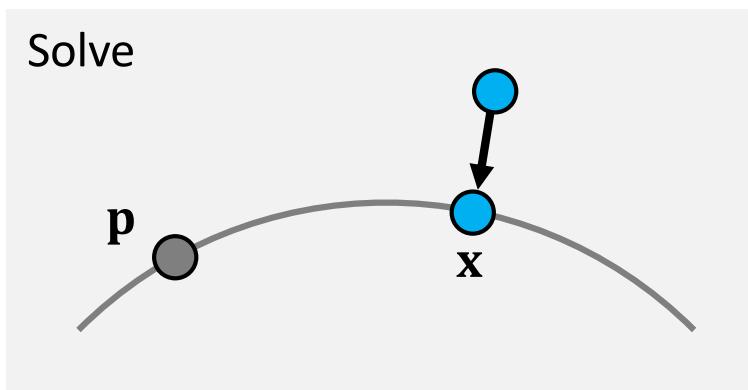
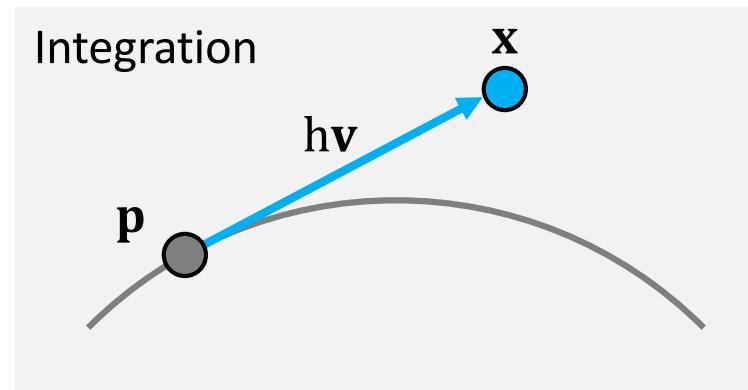
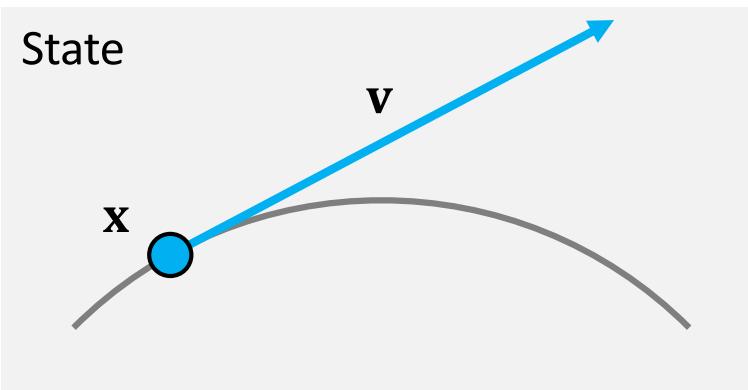
 solve(c , h)

for all particles i

$$\mathbf{v}_i \leftarrow (\mathbf{x}_i - \mathbf{p}_i)/h$$

Spending fixed time budget with sub-steps is much more effective than iterations!

A Bead on a Wire



PDB = integrator **and** solver!

Computing Displacement

- After projection, the constraint should be zero

$$\mathbf{c}(\mathbf{x} + \Delta\mathbf{x}) \approx \mathbf{c}(\mathbf{x}) + \nabla_{\mathbf{x}}\mathbf{c}(\mathbf{x}) \cdot \Delta\mathbf{x} = 0$$

$$\Delta\mathbf{x} = \lambda \nabla_{\mathbf{x}}\mathbf{c}(\mathbf{x})$$

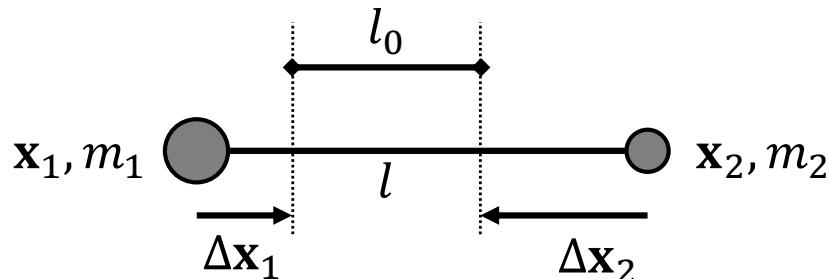
$$\Delta\mathbf{x} = -\frac{\mathbf{c}(\mathbf{x})}{|\nabla_{\mathbf{x}}\mathbf{c}(\mathbf{x})|^2} \nabla_{\mathbf{x}}\mathbf{c}(\mathbf{x})$$

: a.k.a. Newton-Rapson step

- When the constraint is associated with n particles with different masses $m_i = 1/w_i$

$$\Delta\mathbf{x}_i = -\frac{w_i \mathbf{c}(\mathbf{x}_1, \dots, \mathbf{x}_n)}{\sum_j w_j \left| \nabla_{\mathbf{x}_j} \mathbf{c}(\mathbf{x}_1, \dots, \mathbf{x}_n) \right|^2} \nabla_{\mathbf{x}} \mathbf{c}(\mathbf{x}_1, \dots, \mathbf{x}_n)$$

Distance Constraint



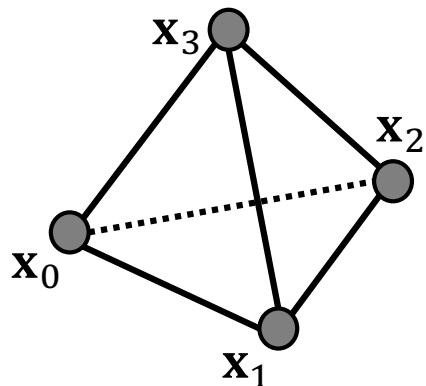
- Rest length l_0
- Current length $l = |\mathbf{x}_1 - \mathbf{x}_2|$
- Masses m_i
- Inverse masses $w_i = 1/m_i$

$$c(\mathbf{x}_1, \mathbf{x}_2) = |\mathbf{x}_1 - \mathbf{x}_2| - l_0$$

$$\frac{\partial c(\mathbf{x}_1, \mathbf{x}_2)}{\partial \mathbf{x}_1} = \frac{\mathbf{x}_1 - \mathbf{x}_2}{|\mathbf{x}_1 - \mathbf{x}_2|} \quad \rightarrow \quad \Delta\mathbf{x}_1 = -\frac{w_1}{w_1 + w_2} (|\mathbf{x}_1 - \mathbf{x}_2| - l_0) \frac{\mathbf{x}_1 - \mathbf{x}_2}{|\mathbf{x}_1 - \mathbf{x}_2|}$$

$$\frac{\partial c(\mathbf{x}_1, \mathbf{x}_2)}{\partial \mathbf{x}_2} = -\frac{\mathbf{x}_1 - \mathbf{x}_2}{|\mathbf{x}_1 - \mathbf{x}_2|} \quad \rightarrow \quad \Delta\mathbf{x}_2 = \frac{w_2}{w_1 + w_2} (|\mathbf{x}_1 - \mathbf{x}_2| - l_0) \frac{\mathbf{x}_1 - \mathbf{x}_2}{|\mathbf{x}_1 - \mathbf{x}_2|}$$

Volume Conservation Constraint



$$c(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = 6(V - V_0)$$

$$c = 6(V - V_0) = [(\mathbf{x}_1 - \mathbf{x}_0) \times (\mathbf{x}_2 - \mathbf{x}_0)] \cdot (\mathbf{x}_3 - \mathbf{x}_0) - 6V_0$$

$$\nabla_{\mathbf{x}_0} c = (\mathbf{x}_3 - \mathbf{x}_1) \times (\mathbf{x}_2 - \mathbf{x}_1)$$

$$\nabla_{\mathbf{x}_1} c = (\mathbf{x}_2 - \mathbf{x}_0) \times (\mathbf{x}_3 - \mathbf{x}_1)$$

$$\nabla_{\mathbf{x}_2} c = (\mathbf{x}_3 - \mathbf{x}_0) \times (\mathbf{x}_1 - \mathbf{x}_0)$$

$$\nabla_{\mathbf{x}_3} c = (\mathbf{x}_1 - \mathbf{x}_0) \times (\mathbf{x}_2 - \mathbf{x}_0)$$



$$\Delta \mathbf{x}_i = \lambda w_i \nabla_{\mathbf{x}_i} c$$

$$\lambda = \frac{-6(V - V_0)}{\sum_j w_j |\nabla_{\mathbf{x}_j} c|^2}$$

Making the Constraint Soft

- PBD:

- Scale the correction as $\Delta \mathbf{x}_i = \mathbf{k} \lambda w_i \nabla_{\mathbf{x}_i} \mathbf{c}$
- Stiffness $\mathbf{k} \in [0, 1]$
- Easy to tune!
- Dependent on time step (stiffer for smaller time steps)

- XPBD:

- Change the Lagrange multipliers as

$$\lambda = \frac{-\mathbf{c}}{\sum_j w_j \left| \nabla_{\mathbf{x}_j} \mathbf{c} \right|^2 + \frac{\alpha}{\Delta t^2}}$$

- Compliance α is the inverse of physical stiffness
- Infinitely stiff (hard) when $\alpha = 0$

Summary

- We have learned how to simulate soft body objects
 - Mass-spring system
 - Measuring deformation using the deformation gradient
 - Position-based formulation
- We haven't covered how to prepare models for simulation (this will not be covered in this course)
 - The quality of discretization largely affects simulation!
 - General rule is to make all elements as regular as possible so that no singularity would happen while numerical calculation. There exist many methods for this